STOR 565 Final Project
The Machine Learners
Portrait Style Transfer

SUMMARY

In this project, we attempted image style transfer using SVD, VGG, and CycleGAN for unpaired image translation. The Singular Value Decomposition (SVD) combines a subject portrait photo with a texture photo to enhance the overall visual interest of a photo portrait. The VGG method maintains details of both content and style images in the generated images and uses a neural network to produce a combined image. Our last approach of CycleGAN is a generative model that trains on two collections of unpaired images representing the desired input and output.

MOTIVATIONS AND DATA

With the release of ChatGPT and popular generative AI art models, ML/AI occupies more and more space in popular media discourse. As a result of the increasing popularity of machine learning-based image generation, we thought this project would give us an important understanding of this technology.

Our project uses three primary data sources:

- Flickr-Faces-HQ (FFHQ): high-quality image dataset of human faces
- Portrait Paintings scrapped from WikiArt
- Pictures of natural landscapes

Images, stored in the .png or .jpg format are represented by a 3-dimensional matrix, where entries correspond to pixels. One dimension is the image width in pixels, one dimension is image height, and the third dimension, of length 3, contains one channel for "red," "green," and "blue" pixel intensity, and, in the case of .png files, a fourth channel for transparency. The SVD model automatically scaled the painting to the same dimensions as the portrait, while The GAN and VGG methods transformed images to have matching dimensions of 256 by 256 pixels.

SVD METHODS

The first of our three models utilize Singular Value Decomposition (SVD) to extract the "subject" from our input photograph and the "texture" from our input portrait, then combines the two to create a stylized portrait. SVD is represented by the equation X = $U\Sigma V^T$, where X is a matrix of data and Σ is a diagonal matrix containing "singular values" obtained via the eigen-decomposition of X. Singular values and their corresponding columns in U and V are ordered according to how much variance they explain from the original X. So, removing right-hand columns from Σ, U, and V reduces the dimensionality of X, while retaining most information, while the removal of left-hand columns results in a high amount of data loss. In fact, with most images in our dataset, we saw that the first singular value contained the vast majority of the photo's variation (Figure 1). However, the small amount of variation left in the remaining singular values is important. Yagmaee et al. show that the first few singular values represent the "structure," or subject, of a painting, while the rest represent the texture (2020). There is no magic number of singular values for this cut-off, nor is there a magic amount of variation: trial and error are necessary for each painting.

As mentioned previously, images are represented by 3-4 2-d color-channel matrices, so for each of these channels the following procedure was applied: take the SVD decomposition of the channel for two images. Extract the structure of the first image by removing *all but* the first few columns of the *Σ,* U, and V matrices. Extract the style of the second image by removing *just* the first few columns. Add these two matrices together. Set all pixels exceeding the maximum possible value (either 1, or 255, depending on the format) to the maximum. The images had to have the same dimensions, so the style image was transformed to match the dimensions of the subject image. Repeating this process for each channel gives an image that has the structure of one image, but the style of the other!

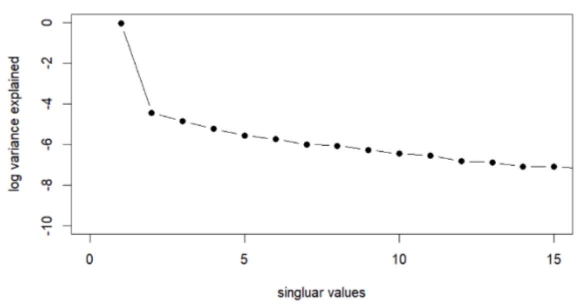


Figure 1. Log variance explained by the first 15 singular values of the SVD decomposition of a Picasso portrait's red color channel.

SVD Results

The SVD model is excellent at removing textures from paintings with thick brushstrokes with few hard edges, such as works of Van Gogh or Monet (Figure 2a). However, the model picks up edges very well around the same singular value range as texture, so paintings lacking distinct texture with hard edges are less universally applicable as style images (Figure 2b). The method is less particular about subject images, however in many cases it is important to match the areas of focus in two images, as no geometric transformations take place, as opposed to some of the neural network models introduced in this paper. For instance, using the Mona Lisa as a style painting for a photograph of a man results in strange shading due to the difference in their silhouette dimensions (Figure 2c).

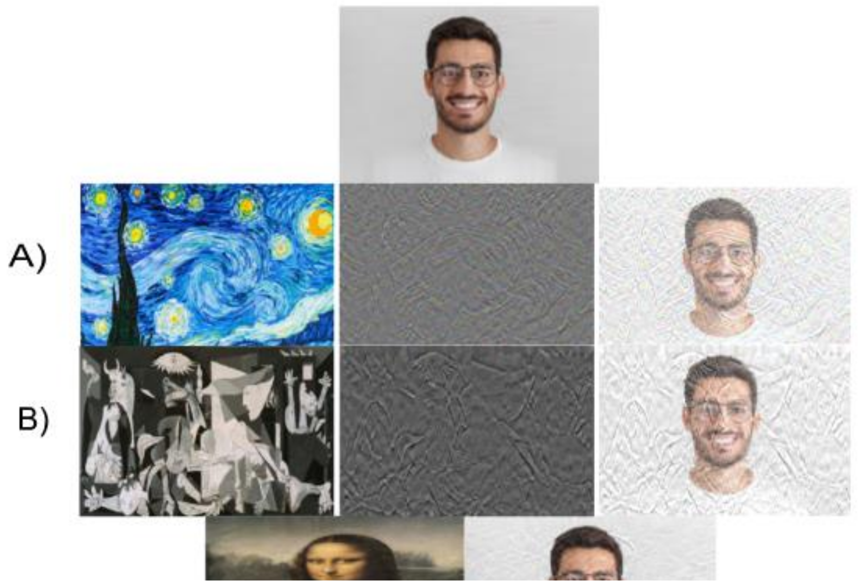


Figure 2 (a,b,c). Removal of the texture layers from Van Gogh's "Starry Night" (a) and Picasso's "Guernica" using the SVD method removing the first 20 singular values from each. Then, these textures, in addition to that of the "Mona Lisa" (c) were applied to a stock photo of a man. Picasso's hard edges make a confusing texture layer, while Van Gogh's flowing brushstrokes create a fairly uniform texture layer, and the mona lisa's silhouette is visible in the final product.

Since this method does not use edge detection to treat faces separately, it can be used for any subject image, given an appropriate style pairing (Figure 3).

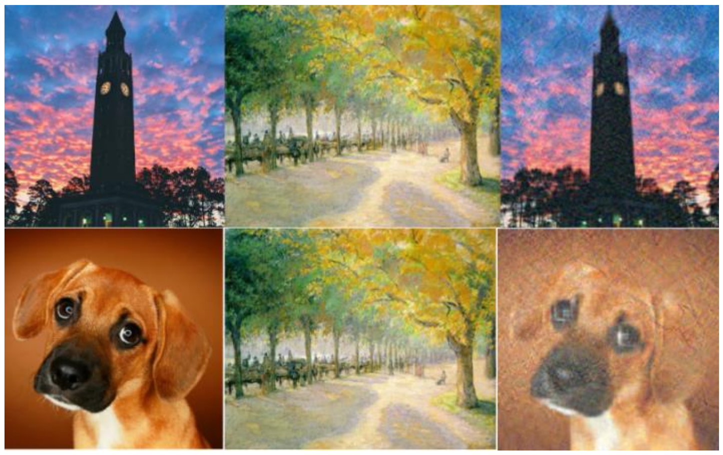


Figure 3. An image of the UNC belltower (top) and a puppy (bottom) and their transformations under the style of Camille Pissaro's 1890 work "Hyde Park, London."

VGG METHOD

The next method is VGG-19. VGG-19 is a pre-trained model using images from the ImageNet database, with over 14,197,122 organized images (Gautam). The component of the model is a convolutional neural network, which is the VGG-19 model itself, the content image, which is the image where we want to transfer style, the style image, the image in which style is to be transferred, and the generated image. There are 19 different layers of the VGG-19 model, hence the 19, and these layers learn different features of the images. The first layer learns the simple textures and the last layer learns the more complex features and patterns (Gautam). Figure 4a shows the different layers' characteristics that the model tries to learn through the 19 different layers, starting from the simplest (edges) to the most complex (objects).

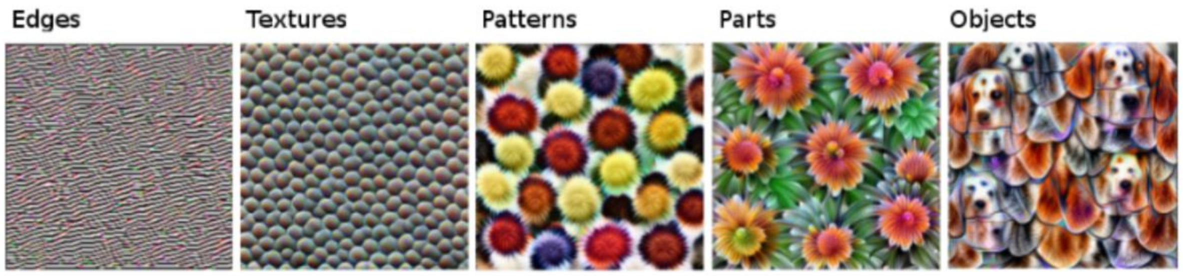


Figure 4a. Aspects of image the model picks up on through various layers

Gautam, Sushant. "Basic Intuition and Guide to Neural Style Transfer." Medium, 8 Jan. 2022, pub.towardsai.net/basic-intuition-on-neural-style-transfer-idea-c5ac179d1530.

The goal of the VGG-19 model is to "maintain details of both content and style images in the generated image" (Gautam). The model goes about doing this by minimizing the "distance" between the style image and the content image. Minimizing the distance between the style image and content image means that the model is minimizing the loss between the two images, an effort to keep features from both of the images. This is utilized through different matrices, the model style matrix and the subject style matrix. We first resized our images to 224 x 224 to standardize, so this means the matrix dimensions (224,224,3). The matrices are then used to compare the RGB contents of the pixels and then the values are minimized between the two matrices (Kaushik). Most neural networks re-train weights based on the input, but the VGG model freezes the weights and reweighs the pixel values instead, as shown in Figure 4b (Gautam).

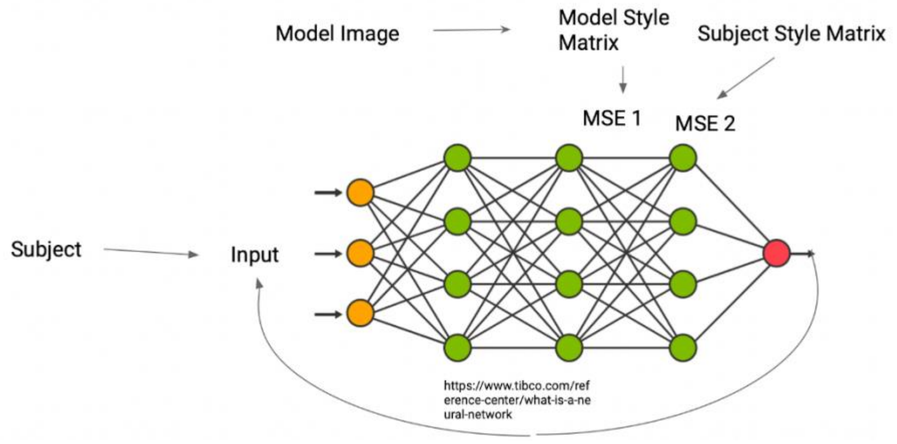


Figure 4b. *Architecture of the VGG-19 Model. Weights are not re-trained and instead frozen, model style matrix and subject style matrix is compared, this process is repeated.*

Our model is slightly different from the original VGG-19 model. The VGG-19 model is used to classify images and when images are run into the model, the images are compared to the style image accordingly and determined if they fit the parameters similar to the style model. Our model differs because it modifies the existing image so that the visual aspects of the image are made similar to the aspects of the style image. It doesn't actually create a new image itself but changes the pixels of the image and keeps going back and forth comparing with the style image to see how similar they are.

First, we looked at outputs after 100 iterations of the model being run, but the results were not what we were looking for. Then we tried running 1000 or more iterations and even though it took much longer to run, the results were much more like what we desired. Figure 5 and 6 shows an instance of the model where 100 iterations were run and an instance where 900 iterations were run.

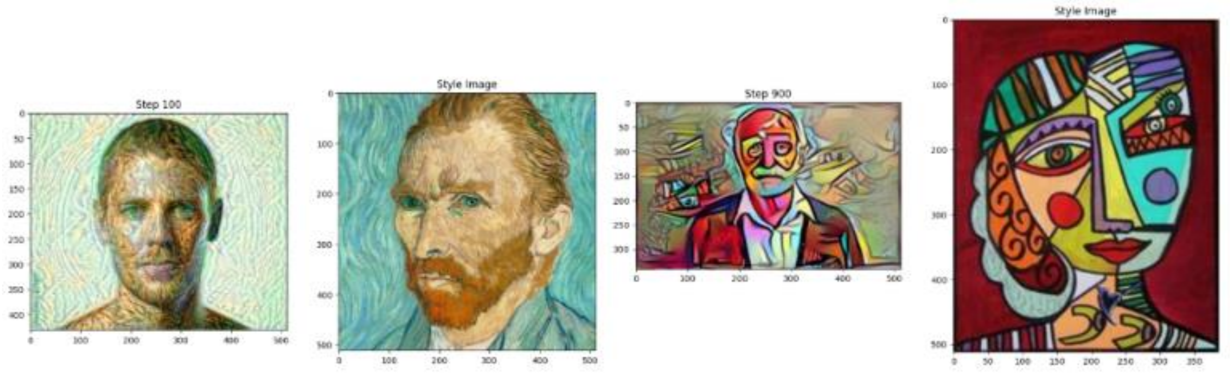


Figure 5 and 6) *Example outputs of VGG-19 model with their respective art style*

At a certain point, the minimum loss between the two images is reached, and more iterations would not make any more changes to the resulting image. Figure 7 shows the progression of the resulting image after 0, 100, 500, and 900 iterations of minimizing loss. The resulting image at step 500 does not differ much from the image at step 900, so we can reasonably infer that the minimum loss is reached before that. However, step 0, step 100, and step 500 present drastically different images, meaning the model is continuing to reach the minimum loss.

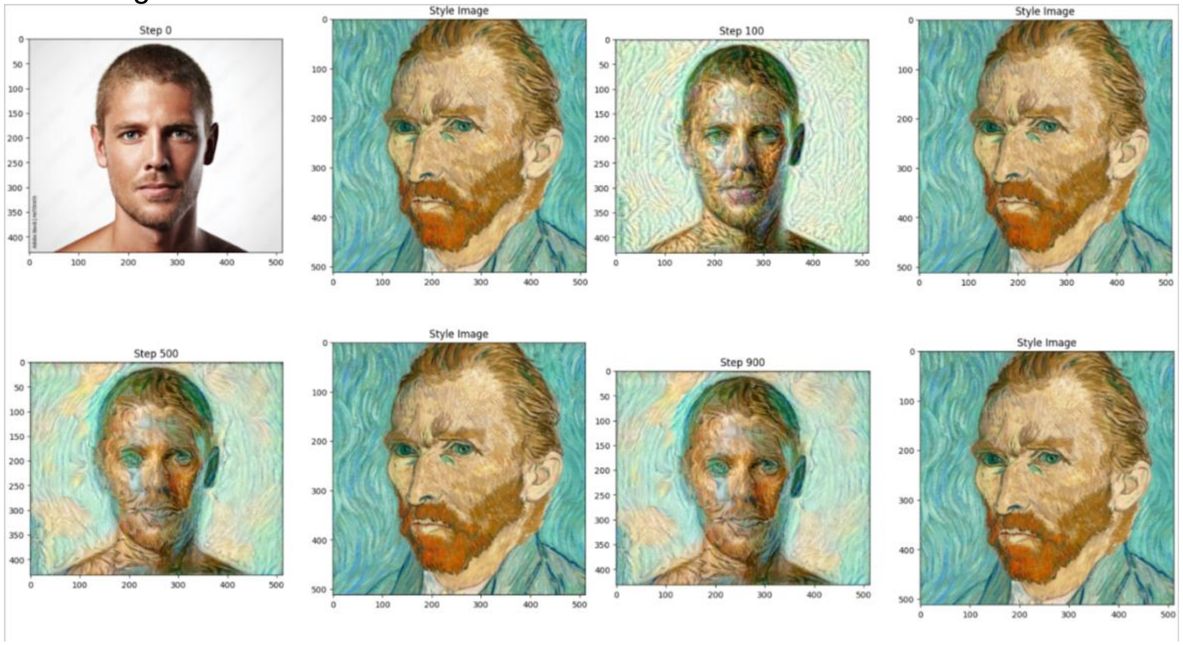


Figure 7. *Progression of the VGG-19 output, through 0, 100, 500, 900 iterations respectively.*

After trying lots of different content images and style images, we discovered that the VGG-19 model is much better at picking up distinctive features or patterns in the style images. Like Figure 5, the model replicates the patterns in the style image and important to note that the model isn't creating a new image itself but modifying the content image to minimize the loss as mentioned before. Figure 8 and 9 show additional examples focusing on the strengths of the VGG-19 model. Overall, we were very satisfied with our results from the VGG-19 model.

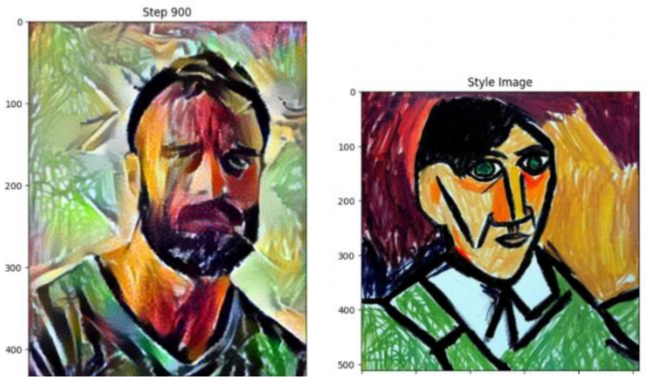


Figure 8. *Additional example of VGG-19 output. VGG-19 is good at replicating patterns of the style image.*

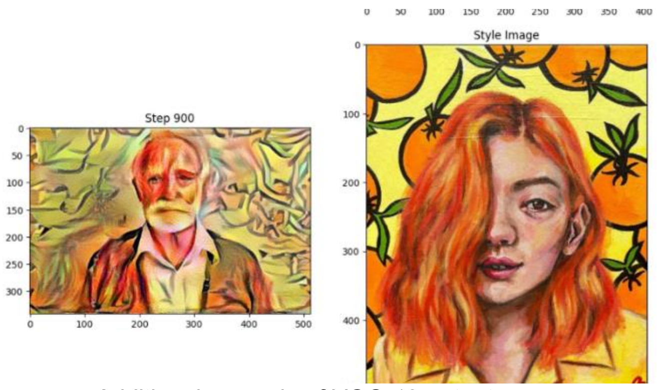


Figure 9. *Additional example of VGG-19 output.*

CYCLEGAN

Our last model is the CycleGAN Model. CycleGAN combines the two concepts of Generative Adversarial Networks (GAN) and Image-to-Image Translation. GANs usually contain two important parts, the generator, and the discriminator. The generator is a model used to generate new examples from the problem's domain. The discriminator on the other hand is used to check these new examples and help determine whether they are real or fake. Image-to-Image Translation, on the other hand, is generally a process where paired datasets that contain the input and output images are used to learn the mapping from input images to output images and use that mapping to generate an output. However, in this case, we have unpaired data that can't be used to map the data (Jun-Yan Zhu).

CycleGAN works around those difficulties by training two generators and two discriminators in a cycle as shown in Figure 10. The cycle begins, in our case, by using the first generator to take a photo and generate a painting. Then our first discriminator takes these generated paintings and the dataset of real paintings to determine whether the generated painting is real or fake. Then the second generator takes the painting and generates a photo, which then the second discriminator takes in along with the dataset of real photos and determines whether the photo is real or fake. Then, it heads back to the first generator to continue the cycle. This cycle is then repeated to train the generators and discriminators to minimize loss.

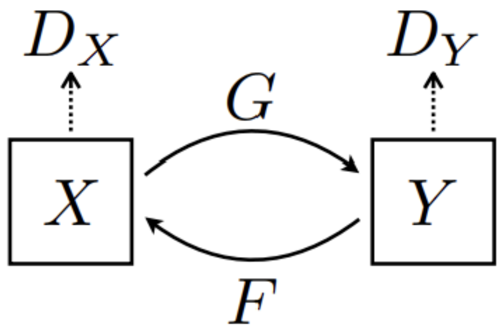

$$\begin{tikzcd} D_X & & D_Y \\ \boxed{X} \arrow[u, dashed] \arrow[rr, bend left, "G"] & & \boxed{Y} \arrow[u, dashed] \arrow[ll, bend left, "F"] \end{tikzcd}$$

Figure 10. Here we have a visual representation of our generators, discriminators, and domains. X and Y are respectively domains for photos and portraits. G and F are the two generator functions which map from photo to portrait and portrait to photo. DX and DY are our two discriminators which determines whether the photo and portrait are real. DX keeps the F mapping function in check and DY keeps the G mapping function in check (Jun-Yan Zhu).

CycleGAN Loss Function

Now we can build our CycleGAN by applying the needed loss functions after defining the generator and discriminator network. We defined the loss function for both the generator network and the discriminator network. The CycleGAN loss functions consist of different components such as adversarial loss, and cycle consistency loss.

Adversarial Loss

This component of the CycleGAN function trains both generator and discriminator. The generator tries to generate an image that the discriminator cannot identify whether the image is real or computer-generated. The discriminator aims to classify correctly and minimize the loss by giving a zero matrix to a computer-generated image and an all-one matrix to a real image (Jun-Yan Zhu). The purpose of adversarial loss is to help the generator to produce realistic images and help the discriminator to increase the accuracy of distinguishing between a real image and a generated image.

Cycle Consistency Loss

This loss function ensures that the image translation is reversible. This basically means the original image should be achieved if the image from domain X translates to domain Y and then translates back to domain X again (Jun-Yan Zhu). This helps the generator to produce consistent mapping and to keep the content of the input image and we quantify by using L1 loss that measures the difference between the input image and the generated image.

$$L1 \;=\; \frac{1}{N} \;\times\; \sum_{1}^{N}(input\ pixels \;-\; generated\ pixels) \qquad\qquad N \;=\; Number\ of\ pixels$$

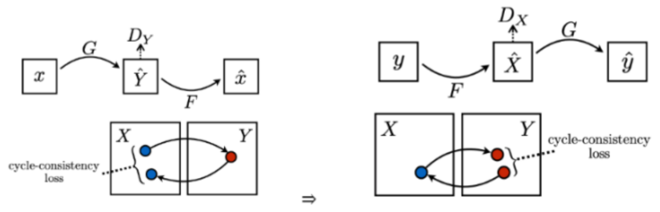
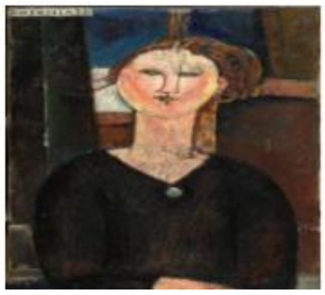


Figure 11 (a,b). 11a: Forward consistency loss: x G(x) F(G(x)) x
11b: Backward consistency loss: y F(y) G(F(y)) y
Jun-Yan Zhu, Taesung Park*, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", in IEEE International Conference on Computer Vision (ICCV), 2017.*

CycleGan results



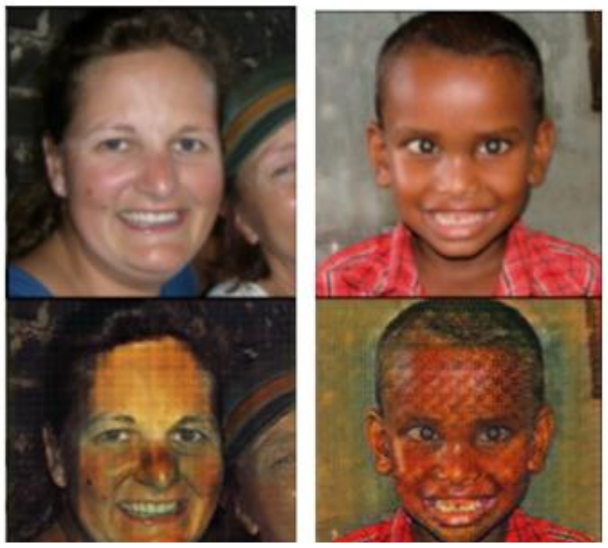


Figure 12. The picture on the left is an example of Amedeo Modigliani art and pictures on the right are results from generating on Modigliani collection of arts

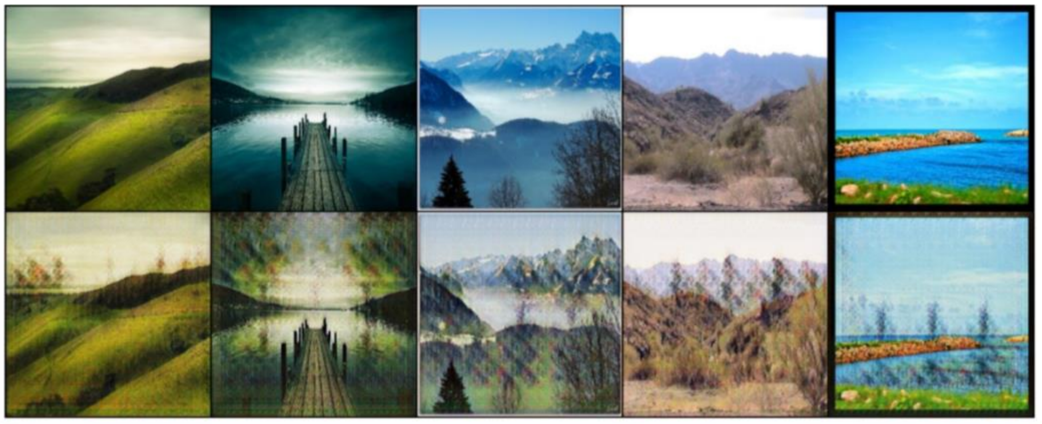


Figure 13. *These pictures generated by CycleGAN using Edvard Munch artwork as training data*

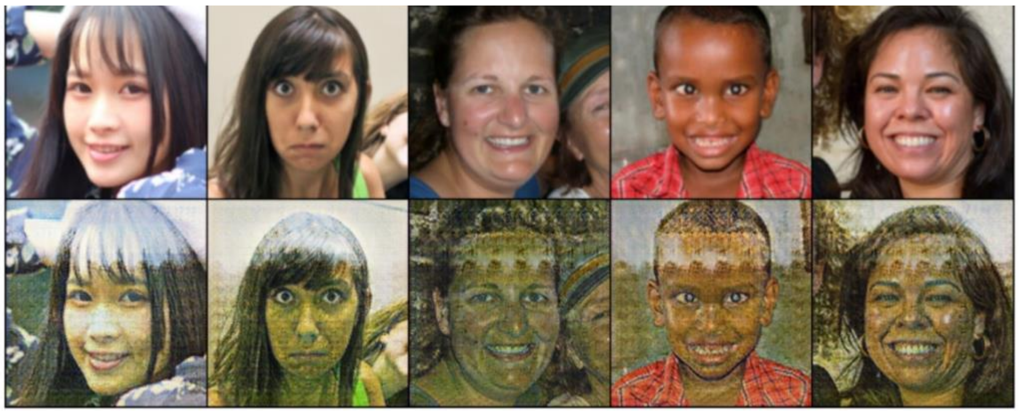


Figure 14. *More pictures generated by CycleGAN using Camille Pissarro artwork as training data*

CONCLUSION

In this project we utilized three methods for image style transfer. SVD transferred texture the best while preserving accuracy of the photo portrait. VGG most accurately combined the style of the images with minimal noise. CycleGAN possessed the most potential for image style translation and captured the subtle shading of various artists styles.

REFERENCES

Gatys L, Ecker A, Bethge M. 2016. A Neural Algorithm of Artistic Style. Journal of Vision. 16(12):326. Doi:https://doi.org/10.1167/16.12.326. https://arxiv.org/pdf/1508.06576.pdf.

Gautam, Sushant. “Basic Intuition and Guide to Neural Style Transfer.” *Medium*, Towards AI, 8 Jan. 2022, https://pub.towardsai.net/basic-intuition-on-neural-style-transfer-idea-c5ac179d1530.

Jun-Yan Zhu*, Taesung Park*, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", in IEEE International Conference on Computer Vision (ICCV), 2017.

Kaushik, Aakash. “Understanding the VGG19 Architecture.” *OpenGenus IQ: Computing Expertise & Legacy*, 26 Feb. 2020, iq.opengenus.org/vgg19-architecture/.

Yaghmaee, F., Peyvandi, K. Improving image inpainting quality by a new SVD-based decomposition. *Multimed Tools Appl* 79, 13795–13809 (2020). https://doi.org/10.1007/s11042-020-08650-x